TOME

NOMBRES COMPLEXES

LES ENSEMBLES DE MANDELBROT ET DE JULIA

Pré-requis : nombres complexes.

PARTIE A : L'ENSEMBLE DE MANDELBROT

On considère la suite complexe (z_n) définie par $z_0=0$ et $z_{n+1}=z_n^2+c$, où $c \in \mathbb{C}$.

L'ensemble de Mandelbrot, que l'on notera $\mathcal{M}$, est : $\mathcal{M} = \{c \in \mathbb{C},\ \lim\limits_{n\to+\infty} |z_n| \neq +\infty\}$.

On peut alors démontrer les deux propriétés suivantes (ce que nous ne ferons pas ici) :

PROPRIÉTÉS

- $|c|>2 \Rightarrow c \notin \mathcal{M}$.
- $\exists\ n_0 \in \mathbb{N},\ |z_{n_0}|>2 \Leftrightarrow c \notin \mathcal{M}$.

REMARQUE : par contraposée de la première propriété, « si $c \in \mathcal{M}$ alors $|c| \leqslant 2$ ».

Et la deuxième propriété donne : $\forall\ n \in \mathbb{N},\ |z_n| \leqslant 2 \Leftrightarrow c \in \mathcal{M}$.

On a donc facilement une autre caractérisation de $\mathcal{M}$:

$$\mathcal{M} = \{c \in \mathbb{C},\ (|z_n|) \text{ est bornée}\}.$$

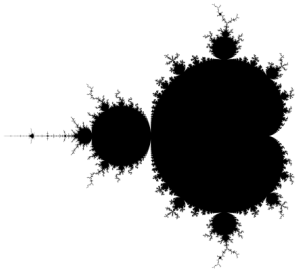
On souhaite écrire un programme, en Python, qui affiche les points d'affixe c qui n'appartiennent pas à $\mathcal{M}$. Pour cela, on décide d'un certain nombre d'itérations que l'on souhaite effectuer avant de conclure que le complexe c n'appartient pas à $\mathcal{M}$. On peut également attribuer une couleur différente selon le nombre d'itérations effectuées avant d'avoir déclaré que c n'appartient pas à $\mathcal{M}$.

On définit des variables qui définiront le domaine dans lequel c varie[1] :

```
xmin, xmax, ymin, ymax = -1.5, 0.5, -1.25, 1.25
```

On définit également le nombre d'itérations avant de conclure sur c. Plus ce nombre sera grand, mieux ce sera mais au prix d'un nombre de calculs qui augmentera… On choisit généralement un entier compris entre 50 et 100.

```
nbiter = 50
```



Écrire un tel programme.

Aide pour représenter un nuage de points :
https://sites.google.com/view/aide-python/graphiques/les-graphiques-courbes-et-nuages-de-points-scatter-plot

Et si vous souhaitez utiliser les complexes dans Python, sachez que i s'écrit `1j`, que `z.real` et `z.imag` renvoient respectivement la partie réelle et la partie imaginaire de `z`.

1 Vous pouvez changer les valeurs mais attention, les calculs sont nombreux dans cet algorithme et cela pourra fortement ralentir la vitesse d'exécution.

Partie B : les ensembles de Julia

On considère la suite complexe (z_n) définie par $z_0 \in \mathbb{C}$ et $z_{n+1}=z_n^2+c$, où $c \in \mathbb{C}$.

L'ensemble de Julia, que l'on notera J, est : $J = \{ z_0 \in \mathbb{C}, \lim\limits_{n\to+\infty} |z_n| \neq +\infty \}$.

Ou encore : $J = \{ z_0 \in \mathbb{C}, (|z_n|) \text{ est bornée} \}$.

On admet que si $c \in$ M, alors $0 \in$ J et J sera représenté géométriquement dans le plan complexe par un ensemble « connexe » (en un seul morceau). Si $c \notin$ M, alors J sera représenté géométriquement par un ensemble de points isolés (de type « poussières de Cantor ») et l'on ne verra rien en l'affichant…

Écrire un programme Python qui affiche l'ensemble de Julia associé à un complexe (aléatoire ou pas).